Geometry of Interaction and Space Complexity

Supervisors: Damiano Mazza (CR CNRS), Thomas Seiller (CR CNRS)
Laboratory: LIPN – UMR 7030 CNRS & Paris 13 University)
Applicant: Sambo Boris Eng

1 Context

Computational complexity. Computational complexity arises from the study of the asymptotic behaviour of computer programs' use of resources, such as time and space. Since a given function (e.g. the characteristic function of the set of prime numbers) can be computed by different programs, researchers have been trying to classify the latter based on the most efficient program that can compute them. Complexity classes are thus defined as sets of functions which can be computed by a program with a given computational complexity. E.g. LOGSPACE is the set of functions computed by a program whose space usage grows (at most) logarithmically w.r.t. the size of the input, while PTIME is the class of functions computed by a program whose execution time is bound by a polynomial in the size of the input. This naturally leads to the **classification problem**, namely understand how complexity classes compare. While many results were obtained, most of them in the early years, they concern very small (e.g. AC_0 and NC_1) and/or very large classes (e.g. PSPACE and EXPSPACE). Many questions, such as LOGSPACE $\stackrel{?}{=}$ PTIME, however remain open after more than fifty years, notwithstanding many research efforts and attempted proof methods.

Descriptive Complexity. Mathematicians have then tried to give characterisations of complexity classes that differ from the original machine-bound definitions, hoping to enable methods from radically different areas of mathematics. Efforts in this direction lead to the development of Descriptive Complexity (DC), a field which studies the types of logics whose individual sentences characterise exactly particular complexity classes. Early developments were the 1974 Fagin-Jones-Selman results [20, 35] characterising the classes NEXPTIME and NPTIME. Many such characterisations have then been given [13, 17, 31] and the method led Immerman to a proof of the celebrated Immerman-Szelepcsényi theorem [34, 58] stating the two complexity classes CONLOGSPACE and NLOGSPACE are equal (though Szelepcsényi's proof does not use logic-based methods).

Implicit Computational Complexity. Implicit Computational Complexity (ICC) develops a related approach whose aim is to study algorithmic complexity only in terms of restrictions of languages and computational principles. It has been established since Bellantoni and Cook' landmark paper [10], and following work by Leivant and Marion [38, 39]. Amongst the different approaches to ICC, several results were obtained by considering syntactic restrictions of *linear logic* [23], a refinement of intuitionnistic logic which accounts for the notion of resources. Linear logic introduces a modality ! marking the "possibility of duplicating" a formula A: the formula A shall be used exactly once, while the formula $!A$ can be used any number of times. Modifying the rules governing this modality then yields variants of linear logic having computational interest: this is how constrained linear logic systems, for instance BLL [29], DLAL [9, 3] and ELL [16], are obtained.

Geometry of Interaction (GOI). Girard initiated the Geometry of Interaction research program [26] to obtain particular kinds of *realisability* models (GOI models) for linear logic [23] accounting for some dynamical aspects (namely that of "cut-elimination"). Through the *Curry-Howard correspondence* [14, 32, 56], GOI models are mathematical models of programs accounting for the *dynamics* of computation. The GOI programme is one of the main inspiration behind game semantics for programming languages, a field which provided answers to long-standing open questions in theoretical computer science, e.g. *full abstraction* [33, 1] and the decidability of higher-order model checking [43].

This line of research also quickly arose as a natural and well-suited tool for the study of computational complexity. Using the first GoI model [25], Abadi, Gonthier and Lévy [30] showed the optimality of Lamping's reduction in lambda-calculus [37] – a model of computation equivalent to Turing machines. It was applied in implicit computational complexity [8], and inspired dal Lago's context semantics [36] which, among other things, was used to prove complexity bounds on subsystems of System T [15] and linear logic [45, 46]. More recently the GoI program inspired new techniques in implicit computational complexity. Initiated by Girard [27], they have known a rapid development, leading to a series of results in the form of new characterisations of the classes coNLogspace[1] [27, 6], Logspace [7, 4] and Ptime [5].

Interaction Graphs and a Semantic Approach to Complexity Among the most recent and full-fledged embodiement of this program lie Seiller's Interaction Graphs models [48, 51, 52, 53]. These models, in which proofs/programs are interpreted as *graphings* – generalisations of dynamical systems –, encompass all previous GoI models introduced by Girard [53]. In particular, Interaction Graphs allow for modelling quantitative features of programs/proofs [52].

Based on a study of Interaction Graphs models characterising complexity classes [54, 55], Seiller has proposed to use graphings to develop a semantic approach to complexity theory [50]. The intuition behind this program is to model and study programs as dynamical systems that act on a space thought of as the space of configurations. As dynamical systems are inherently deterministic, the use of graphings is needed to extend the approach to probabilistic and/or non-deterministic programs. One can then study a program through the geometry of the associated graphing (for instance, a configuration caught in a loop is a point of the space of finite orbit).

Transcendental Syntax. In a recent series of papers and preprints [28, 21, 22], Girard proposed a new framework fulfilling the geometry of interaction program. It is built on notions from term rewriting, providing a more syntactic approach to GoI models, and thus more concrete and tractable than previous instances based on involved mathematical theories (e.g. operator algebras, ergodic theory). This framework, called *transcendental syntax*, also refines former work by proposing a better treatment of some logical aspects of the models. In particular, the last preprint [22] explains how to extend the approach to first order quantification. This novelty and the associated treatment of equality, is of particular interest, but Girard only sketches the definitions, even the basic ones, and no proofs are provided.

2 Objectives

The goal of the thesis is to use geometry of interaction techniques, and in particular the new tools provided by Girard's most recent model (transcendental syntax), to characterise and provide new insights on space complexity classes. We identify two main tasks for this purpose, detailed below.

Transcendental syntax and Complexity. The core objective of this first task is to formalise and prove correct the constructions of Girard's transcendental syntax, and in particular reach a full understanding of the novel account of predicate calculus it proposes. As part of this task, the connections between this framework and other geometry of interaction models, such as Seiller's Interaction Graphs, will be established in order to fully understand what new tools and methods are made available.

Based on this, the semantic account of computational complexity proposed by Seiller in the Interaction Graphs models [49, 50, 54, 44] will be transferred to the more syntactic model. This will in particular extend the characterisations of complexity classes in terms of prefix-rewriting and/or logic programming obtained by Aubert, Bagnol, Pistone and Seiller [7, 4, 5]. Moreover, the account of predicate calculus could provide grounds for establishing a direct connection with results in Descriptive Complexity, in particular the characterisations of sub-polynomial complexity classes such as AC_0, Logspace and NLogspace.

The applicant, Sambo Boris Eng, already started working on transcendental syntax as part of his internship under the supervision of Thomas Seiller. This allowed him to familiarise himself with the technical background, and formalise the construction of a restricted fragment of linear logic.

[1] Though it is known that coNLogspace = NLogspace [34, 58], the characterisation is naturally one of coNLogspace.

Space complexity. It is well known that the usual complexity measures of time and space (defined using Turing machines) are strongly related to *size* and *depth* of Boolean circuits. In particular, time is polynomially related to circuit *size* of circuits and space is linearly related to circuit *depth*.

Our purpose is to investigate what happens in the higher order world, which is the preferred setting for the theory of programming languages. In particular:

(a) time and space for Turing machines must be replaced by notions of time and space complexity for λ-terms;

(b) a suitable notion of higher-order Boolean circuit must be introduced, and the corresponding notions of size and depth defined for it.

For what concerns time, one may resort to the number of head-reduction steps, which was proved by Accattoli and Dal Lago [2] to be an invariant cost measure, *i.e.*, it is polynomially related to execution time on Turing machines.

When it comes to space complexity, however, much less is known. The ideal would be to find an abstract measure, something that is as much as possible machine-independent. Resorting to notions which are purely internal to the theory of the λ-calculus should guarantee a good level of abstraction (counting head-reductions is a perfect example). The work of Blelloch et al. [57] is a very interesting proposal, although it still relies on a low-level description of λ-terms, which is not entirely satisfactory from our point of view.

Let us temporarily ignore the issue and let us address directly point (b): what is a higher-order Boolean circuit? We believe that this should be taken to coincide with *affine λ-terms*, *i.e.*, programs in which duplication of higher-order data is forbidden. The relationship between affine λ-terms and general λ-terms may be refined by formalizing the intuition (already present in [24]) that the affine λ-calculus is "dense" in the full λ-calculus. This is the object of [40], which may be informally summarized as follows:

- there is a notion of *affine approximation* $t \sqsubset M$ of a λ-term by affine terms t, such that M may be seen as the limit of its affine approximations;
- reduction is continuous: if $M \to^* N$, then for all $u \sqsubset N$ there exists $t \sqsubset M$ such that $t \to^* u$.

The above notion of approximation allows to reconstruct, in the context of the λ-calculus, the same relatioship time/size that exists for Turing machines/Boolean circuits [41], which leads us to postulate the "equation"

$$\frac{\text{affine } \lambda\text{-terms}}{\lambda\text{-terms}} = \frac{\text{Boolean circuits}}{\text{Turing machines}}$$

The goal of this part of the thesis is to complete the picture by finding the analogue of the space/depth relationship for the λ-calculus. Our approach is based on an interesting correspondence between intersection types and affine approximations, found recently by Mazza and his former Ph.D. stidents [42]: a λ-term M has an intersection type iff there exists a simply-typable affine term t such that $t \sqsubset M$. So, we may write $\vdash t \sqsubset M : A$ to say that M is typable (in intersection types) of type A with a derivation isomorphic to t (whose simple type is also A). Exploiting this correspondence and the idea, originally due to Schöpp [47], that the geometry of interaction may be used to perform space-efficient computation on λ-terms, Eng, during his M1 internship with Mazza, proved that one may infer the space complexity of the function computed by a λ-term M from the syntactic depth (as trees) of the intersection types for M.

The above result suggests that, if we take higher-order Boolean circuits to be affine terms of type $\mathsf{Str}[] \multimap \mathsf{Bool}$ (with $\mathsf{Str}[]$ some instance of the type of binary strings), then we should take as "depth" the depth of the formula $\mathsf{Str}[]$. This is in accord with what Terui did for Boolean proof nets [59] and agrees with what Borodin [12] found for Turing machines and (first-order) Boolean circuits. This may also allow us to bypass the definition of a space measure for parsimonious λ-terms: the existence of a family of suitable typings would constitute *per se* an abstract, machine-independent measure. The relationship between intersection types and time complexity of λ-terms was well known [19, 11, 60, 18]; it is interesting to see that it may work for space, too. What remains to be seen, and it will be one of the objectives of the thesis, is whether this measure is space-invariant with respect to Turing machines.

References

[1] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. In *TACS '94: Proceedings of the International Conference on Theoretical Aspects of Computer Software*, number 789 in Lecture

Notes in Computer Science, pages 1–15, London, UK, 1994. Springer-Verlag.

[2] B. Accattoli and U. Dal Lago. On the invariance of the unitary cost model for head reduction. In *Proceedings of RTA*, pages 22–37, 2012.

[3] V. Atassi, P. Baillot, and K. Terui. Verification of ptime reducibility for system f terms via dual light affine logic. In Z. Ésik, editor, *Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 150–166. Springer Berlin Heidelberg, 2006.

[4] C. Aubert, M. Bagnol, P. Pistone, and T. Seiller. Logic programming and logarithmic space. In J. Garrigue, editor, *Programming Languages and Systems - 12th Asian Symposium, APLAS 2014, Singapore, November 17-19, 2014, Proceedings*, volume 8858 of *Lecture Notes in Computer Science*, pages 39–57. Springer, 2014.

[5] C. Aubert, M. Bagnol, and T. Seiller. Unary resolution: Characterizing ptime. In *FOSSACS 2016*, 2016.

[6] C. Aubert and T. Seiller. Characterizing co-nl by a group action. *Mathematical Structures in Computer Science*, 26:606–638, 2016.

[7] C. Aubert and T. Seiller. Logarithmic space and permutations. *Information and Computation*, 248:2–21, 2016.

[8] P. Baillot and M. Pedicini. Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2):1–31, 2001.

[9] P. Baillot and K. Terui. Light types for polynomial time computation in lambda calculus. *Information and Computation*, 207(1):41 – 62, 2009.

[10] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2, 1992.

[11] A. Bernadet and S. Lengrand. Complexity of strongly normalising lambda-terms via non-idempotent intersection types. In *Proceedings of FOSSACS*, pages 88–107, 2011.

[12] A. Borodin. On relating time and space to size and depth. *SIAM J. Comput.*, 6(4):733–744, 1977.

[13] K. J. Compton and E. Grädel. Logical definability of counting functions. *J. Comput. Syst. Sci.*, 53, 1996.

[14] H. B. Curry. Functionality in combinatory logic. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 20, pages 584–590, Nov. 1934.

[15] U. dal Lago. Context semantics, linear logic, and computational complexity. *ACM Transactions on Computational Logic*, 10(4), 2009.

[16] V. Danos and J.-B. Joinet. Linear logic & elementary time. *Information and Computation*, 183(1):123–137, 2003.

[17] A. Dawar and E. Grädel. The descriptive complexity of parity games. *LMCS*, 5213, 2008.

[18] E. De Benedetti and S. Ronchi Della Rocca. Bounding normalization time through intersection types. In *Proceedings of ITRS*, pages 48–57, 2013.

[19] D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009.

[20] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *SIAM-AMS Proc.*, volume 7, 1974.

[21] J.-Y. Girard. Transcendental syntax ii: non deterministic case. Logical Methods in Computer Science (to appear).

[22] J.-Y. Girard. Transcendental syntax iii: equality. Manuscript.

[23] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

[24] J.-Y. Girard. Linear logic. *Theor. Comput. Sci.*, 50(1):1–102, 1987.

[25] J.-Y. Girard. Geometry of interaction I: Interpretation of system F. In *In Proc. Logic Colloquium 88*, 1989.

[26] J.-Y. Girard. Towards a geometry of interaction. In *Proceedings of the AMS Conference on Categories, Logic and Computer Science*, 1989.

[27] J.-Y. Girard. Normativity in logic. In P. Dybjer, S. Lindström, E. Palmgren, and G. Sundholm, editors, *Epistemology versus Ontology*, volume 27 of *Logic, Epistemology, and the Unity of Science*, pages 243–263. Springer, 2012.

[28] J.-Y. Girard. Transcendental syntax i: deterministic case. *Mathematical Structures in Computer Science*, 27(5):827–849, 2017.

[29] J.-Y. Girard, A. Scedrov, and P. J. Scott. Bounded linear logic: a modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, Apr. 1992.

[30] G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In R. Sethi, editor, *POPL*, pages 15–26. ACM Press, 1992.

[31] E. Grädel and Y. Gurevich. Tailoring recursion for complexity. *Journal of Symbolic Logic*, 60, 1995.

[32] W. A. Howard. The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980.

[33] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.

[34] N. Immerman. Nondeterministic space is closed under complementation. In *Structure in Complexity Theory Conference*, 1988.

[35] N. Jones and A. Selman. Turing machines and the spectra of first-order formulas. *Journal of Symbolic Logic*, 39, 1974.

[36] U. d. Lago. The geometry of linear higher-order recursion. *ACM Trans. Comput. Logic*, 10(2):8:1–8:38, Mar. 2009.

[37] J. Lamping. An algorithm for optimal lambda calculus reduction. In F. E. Allen, editor, *POPL*, pages 16–30. ACM Press, 1990.

[38] D. Leivant and J.-Y. Marion. Lambda calculus characterizations of poly-time. *Fundam. Inform.*, 19, 1993.

[39] D. Leivant and J.-Y. Marion. Ramified recurrence and computational complexity II: Substitution and poly-space. *Lecture Notes in Computer Science*, 933, 1994.

[40] D. Mazza. An infinitary affine lambda-calculus isomorphic to the full lambda-calculus. In *Proceedings of LICS*, pages 471–480, 2012.

[41] D. Mazza. Church meets cook and levin. In *Proceedings of LICS*, pages 827–836, 2016.

[42] D. Mazza, L. Pellissier, and P. Vial. Polyadic approximations, fibrations and intersection types. *Proceedings of the ACM on Programming Languages*, 2(POPL:6), 2018.

[43] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 81–90, 2006.

[44] L. Pellissier and T. Seiller. Prams over integers do not compute maxflow efficiently. submitted, 2018.

[45] M. Perrinel. *Investigating the expressivity of linear logic subsystems characterizing polynomial time*. PhD thesis, Ecole normale supérieure de lyon - ENS LYON, 2015.

[46] M. Perrinel. Paths-based criteria and application to linear logic subsystems characterizing polynomial time. *Information and Computation*, 261, 2018.

[47] U. Schöpp. Space-efficient computation by interaction. In *Proceedings of CSL*, pages 606–621, 2006.

[48] T. Seiller. Interaction graphs: Multiplicatives. *Annals of Pure and Applied Logic*, 163:1808–1837, December 2012.

[49] T. Seiller. Measurable preorders and complexity. Topology, Algebra and Categories in Logic Conference, 2015.

[50] T. Seiller. Towards a *Complexity-through-Realizability* theory. http://arxiv.org/pdf/1502.01257, 2015.

[51] T. Seiller. Interaction graphs: Additives. *Annals of Pure and Applied Logic*, 167:95 – 154, 2016.

[52] T. Seiller. Interaction graphs: Full linear logic. In *IEEE/ACM Logic in Computer Science (LICS)*, 2016.

[53] T. Seiller. Interaction graphs: Graphings. *Annals of Pure and Applied Logic*, 168(2):278–320, 2017.

[54] T. Seiller. Interaction graphs: Nondeterministic automata. *ACM Transaction in Computational Logic*, 19(3), 2018.

[55] T. Seiller. Interaction graphs: Probabilistic automata. In preparation, 2019.

[56] M. H. Sørensen and P. Urzyczyn. *Lectures on the Curry-Howard Isomorphism, Volume 149 (Studies in Logic and the Foundations of Mathematics)*. Elsevier Science Inc., New York, NY, USA, 2006.

[57] D. Spoonhower, G. E. Blelloch, R. Harper, and P. B. Gibbons. Space profiling for parallel functional programs. *J. Funct. Program.*, 20(5-6):417–461, 2008.

[58] R. Szelepscényi. The method of forced enumeration for nondeterministic automata. *Bulletin of the EATCS*, 33, 1987.

[59] K. Terui. Proof nets and boolean circuits. In *Proceedings of LICS*, pages 182–191, 2004.

[60] K. Terui. Semantic evaluation, intersection types and complexity of simply typed lambda calculus. In *Proceedings of RTA*, pages 323–338, 2012.